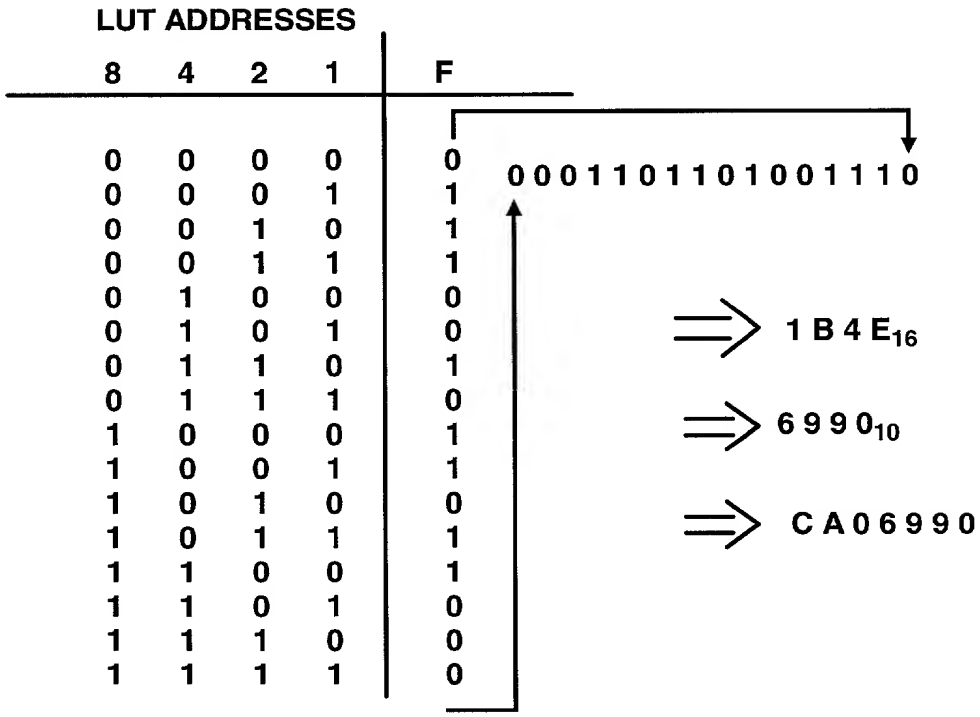
LUT ADDRESSES
8 4 2 1 F
0 0 0 0 0
0 0 0 1 1
0 0 1 0 1
0 0 1 1 1
0 1 0 0 0
0 1 0 1 0
0 1 1 0 1
0 1 1 1 0
1 0 0 0 1
1 0 0 1 1
1 0 1 0 0
1 0 1 1 1
1 1 0 0 1
1 1 0 1 0
1 1 1 0 0
1 1 1 1 0
0 0 0 1 1 0 1 1 0 1 0 0 1 1 1 0
⇒ 1 B 4 E16
⇒ 6 9 9 0 10
⇒ C A 0 6 9 9 0

FIG 1A

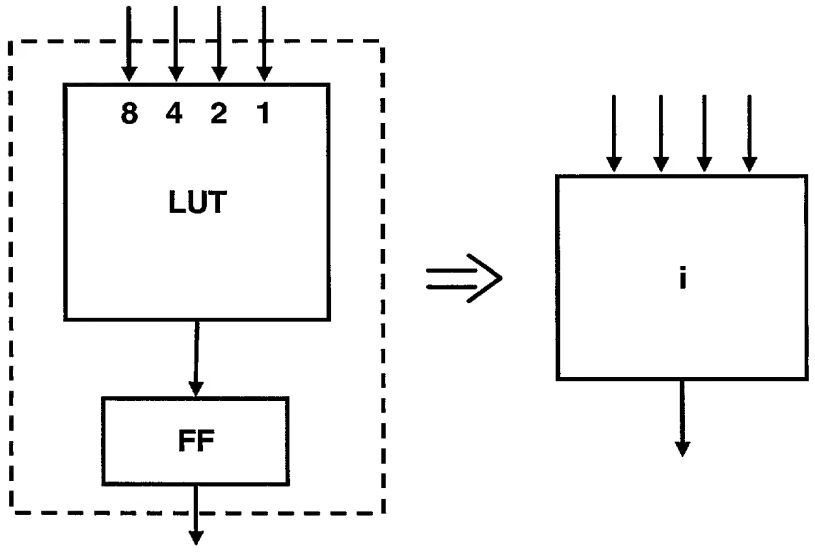
8 4 2 1
LUT
FF
⇒
i

FIG 1B

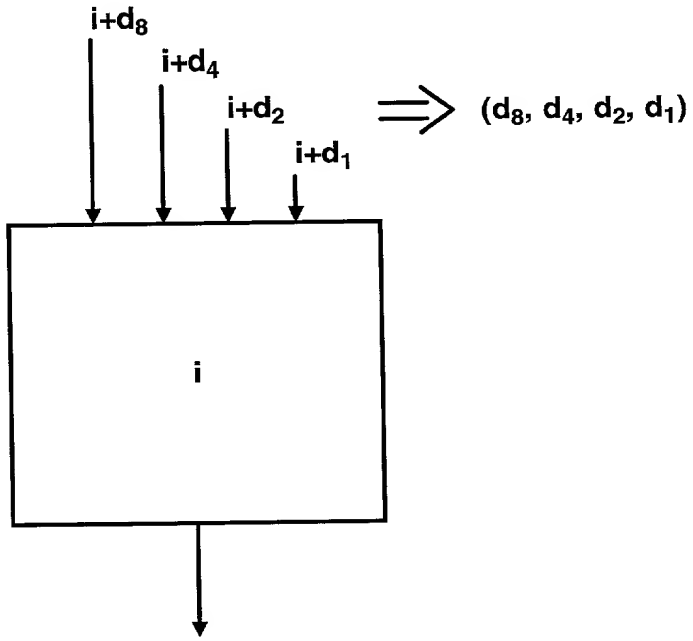

FIG 1C

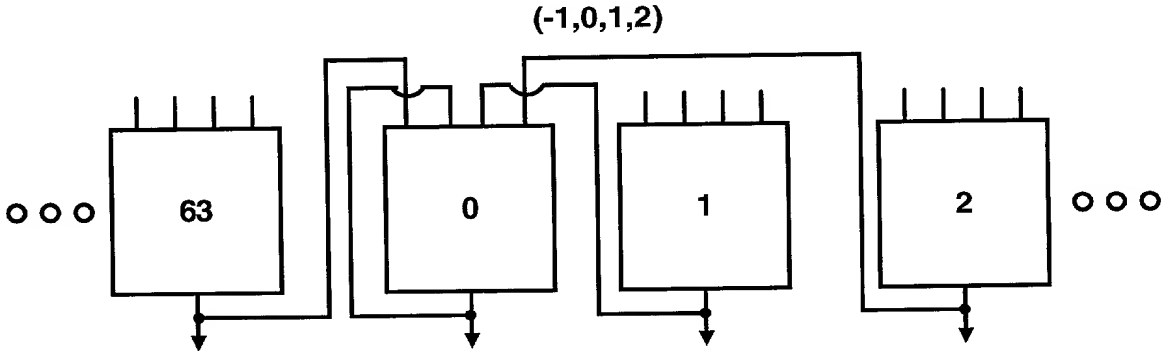

FIG 1D

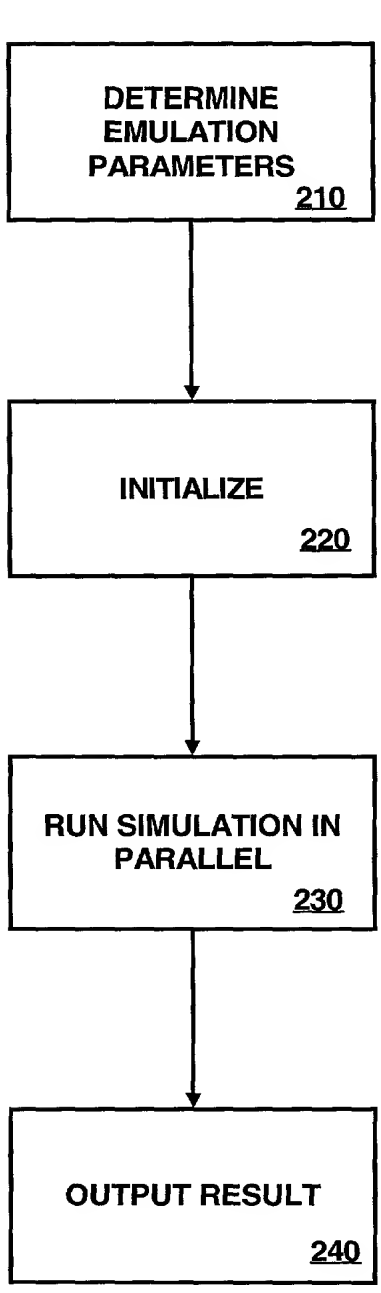
200
DETERMINE EMULATION PARAMETERS
210
INITIALIZE
220
RUN SIMULATION IN PARALLEL
230
OUTPUT RESULT
240

FIG 2

300

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/uio.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>

#ifndef BUFSIZ
#define BUFSIZ 4096
#endif /* BUFSIZ */
#define WSIZE 32
#define scmp(a,b) (strcmp((a),(b)) == 0)
#define rotateLeft(a,b) ((a<<b)|(a>>(WSIZE-b)))

/* Logic for TruthTable 51510 */
#define f(a8,a4,a2,a1) ((a8 & a4 & a2 & a1 )|(a8 & a4 & a2 & ~a1 )|(a8 & ~a4
& a2 & a1 )|(a8 & ~a4 & ~a2 & ~a1 )|(~a8 & a4 & ~a2 & a1 )|(~a8 & a4 & ~a2 &
~a1 )|(~a8 & ~a4 & a2 & ~a1 )|(~a8 & ~a4 & ~a2 & a1 ))

static unsigned long stateEven, stateOdd ;
unsigned long *returnState = &stateEven ;

char *outFile = "bsrand.32" ;
unsigned long initUpper = 0x80000000 ;
unsigned long initLower = 0x00000000 ;
long initRun = 64 ;
long numData = 3000000 ;

void bsinit(unsigned long upper, unsigned long lower)
{
      int i ;
      stateEven = 0 ;
      stateOdd  = 0 ;
      for ( i = 0 ; i < 32 ; i += 2 ) {
            stateEven <<= 1 ;
            stateOdd  <<= 1 ;
            stateEven |= (upper >> (31 - i)) & 0x01 ;
            stateOdd  |= (upper >> (30 - i)) & 0x01 ;
      }
      for ( i = 0 ; i < 32 ; i += 2 ) {
            stateEven <<= 1 ;
            stateOdd  <<= 1 ;
            stateEven |= (lower >> (31 - i)) & 0x01 ;
            stateOdd  |= (lower >> (30 - i)) & 0x01 ;
      }
}

/* { -7, 0, 11, 17 } */
unsigned long bsrand()
{
      unsigned long newStateEven ;
      newStateEven = f(
            rotateLeft(stateOdd , 28),
            stateEven,
            rotateLeft(stateOdd ,  5),
            rotateLeft(stateOdd ,  8)
      ) ;
      stateOdd = f(
            rotateLeft(stateEven, 29),
            stateOdd,
            rotateLeft(stateEven,  6),
```

FIG 3A

```
            rotateLeft(stateEven,  9)
      ) ;
      stateEven = newStateEven ;
      return( *returnState ) ;                                    300
}

void usage(char *name)
{
      fprintf(stderr, "usage: %s ", name) ;
      fprintf(stderr, "[-o outFileName] [-seed upper lower] ") ;
      fprintf(stderr, "[-even | -odd] [-help] ") ;
      fprintf(stderr, "[-num numData] [-skip numInitialRun]\n") ;
}

void getOpt(int argc, char *argv[])
{
      while(argc > 1) {
            if ((argc > 2)&&scmp(argv[1], "-o")) {
                  outFile = argv[2] ;
                  argc -= 2 ; argv += 2 ;
            } else if ((argc > 2)&&scmp(argv[1], "-num")) {
                  sscanf(argv[2], "%ul", &numData) ;
                  argc -= 2 ; argv += 2 ;
            } else if ((argc > 2)&&scmp(argv[1], "-skip")) {
                  sscanf(argv[2], "%ul", &initRun) ;
                  argc -= 2 ; argv += 2 ;
            } else if ((argc > 3)&&scmp(argv[1], "-seed")) {
                  sscanf(argv[2], "%ul", &initUpper) ;
                  sscanf(argv[3], "%ul", &initLower) ;
                  argc -= 3 ; argv += 3 ;
            } else if (scmp(argv[1], "-odd")) {
                  returnState = &stateOdd ;
                  argc-- ; argv++ ;
            } else if (scmp(argv[1], "-even")) {
                  returnState = &stateEven ;
                  argc-- ; argv++ ;
            } else {
                  usage( argv[0] ) ;
                  exit( 1 ) ;
            }
      }
}

int main(int argc, char *argv[])
{
      int fd ;
      unsigned long num, i, j, unit ;
      unsigned long *buf ;
      unit = BUFSIZ / sizeof( num ) ;
      buf = (unsigned long*)malloc( BUFSIZ ) ;
      getOpt( argc, argv ) ;
      fd = open( outFile, O_WRONLY | O_CREAT, 0666 ) ;
      bsinit( initUpper, initLower ) ;
      for ( i = 0 ; i < initRun ; i++ ) {
            bsrand() ;
      }
      for ( i = 0, j = 0  ; i < numData ; i++ ) {
            buf[j++] =  bsrand() ;
            if ( j == unit ) {
                  write(fd, buf, BUFSIZ) ;
                  j = 0 ;
```

FIG 3B

```
            }
        }                                                    300
        close( fd ) ;
}
Software Implementation of Cellular Automata
Based Random Number Generator
J. Barry SHACKLEFORD et al.
HP Docket No. 10019023-1
```


FIG 3C

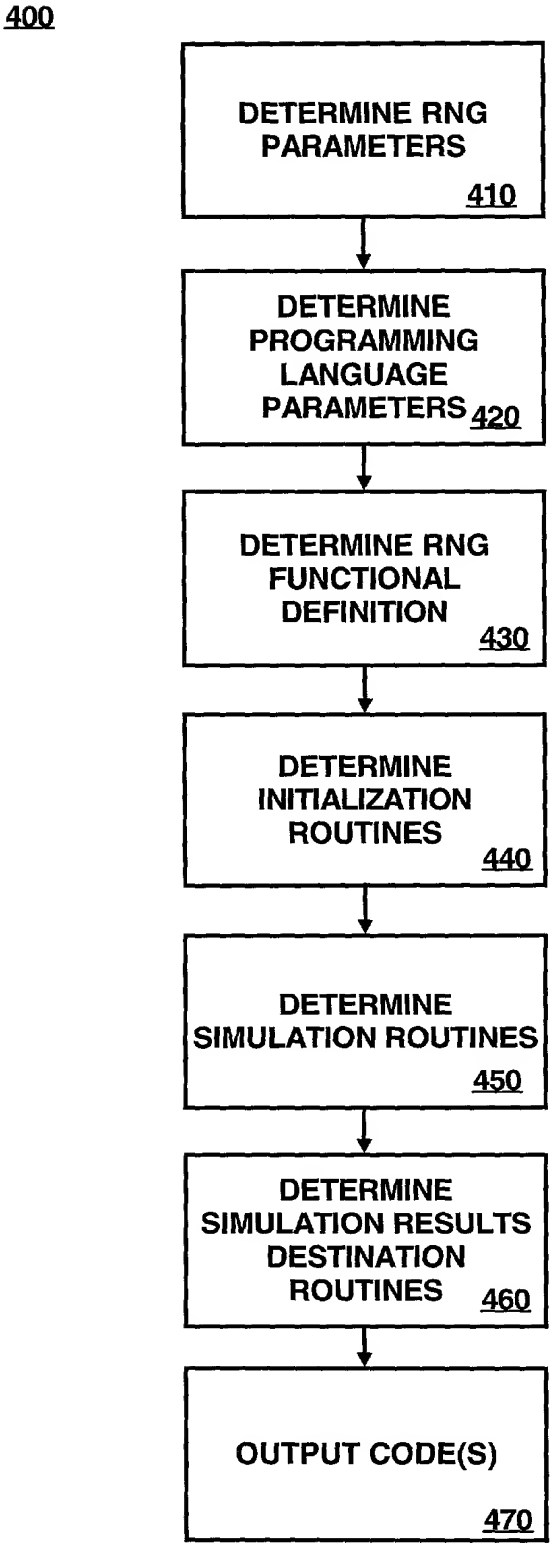
400
DETERMINE RNG PARAMETERS
410
DETERMINE PROGRAMMING LANGUAGE PARAMETERS
420
DETERMINE RNG FUNCTIONAL DEFINITION
430
DETERMINE INITIALIZATION ROUTINES
440
DETERMINE SIMULATION ROUTINES
450
DETERMINE SIMULATION RESULTS DESTINATION ROUTINES
460
OUTPUT CODE(S)
470

FIG 4

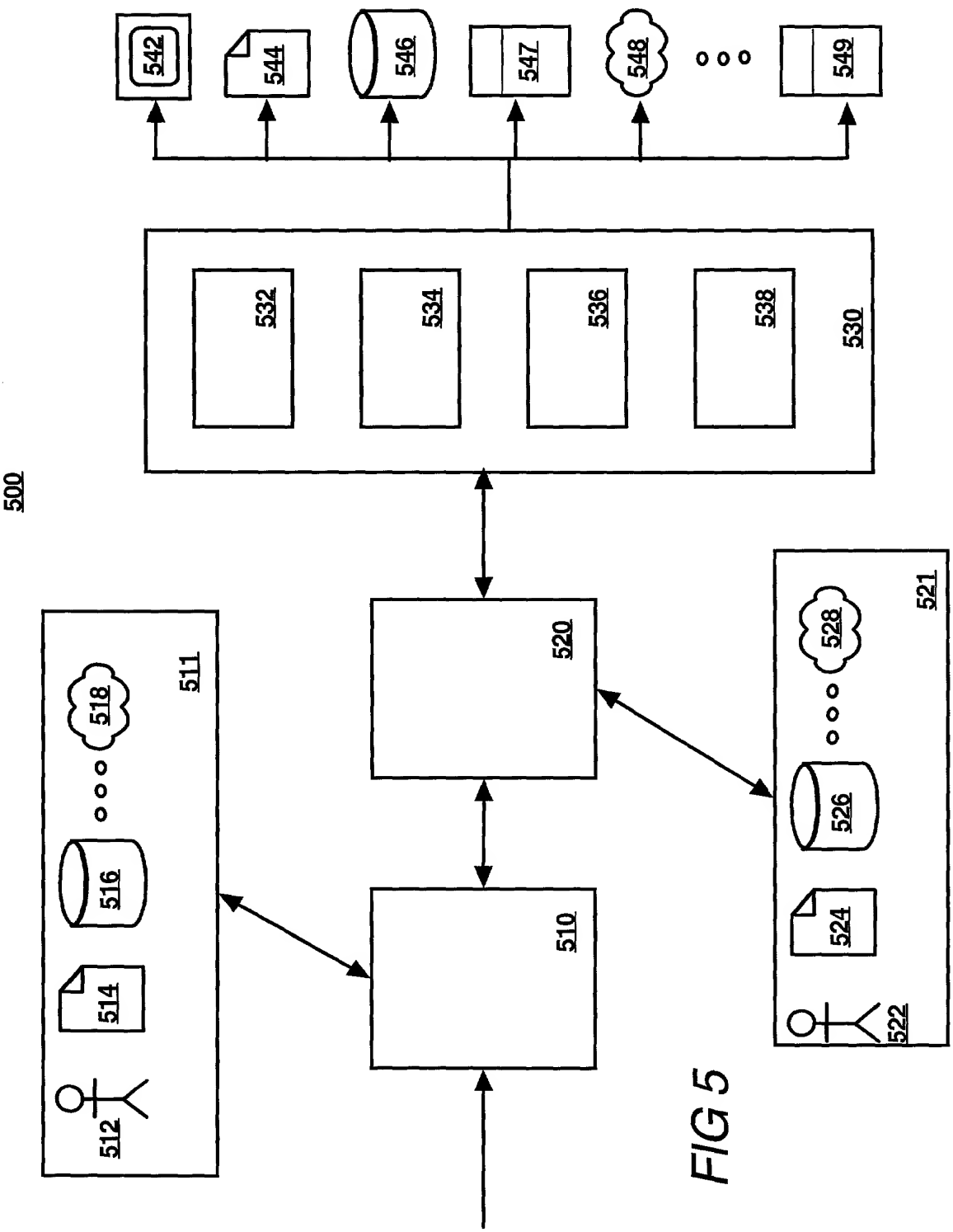
500
542
544
546
547
548
549
532
534
536
538
530
520
511
518
516
514
512
510
521
528
526
524
522

FIG 5

600

```
/*****************************************
 * mkbsrand.c
 *
 * make Barry Shackleford's rand generator program.
 * $Header: /home/moto/work/FPGA2002/DIEHARD/src/MKBSRAND/RCS/mkbsrand.c,v
1.2 2001/09/07 21:26:47 moto Exp moto $
 *
 * Created: 2-Jul-2001
 * Modified: $Date: 2001/09/07 21:26:47 $
 * Author: Motoo Tanaka
 * Copyright (c) Hewlett Packard Company
 *****************************************/

#include <stdio.h>
unsigned int truthTable = 0 ;
int a[4], d[4] ;
int isFirst = 1 ;
FILE *ofp ;

FILE *efopen(char *filename, char *mode)
{
      FILE *fp ;
      fp = fopen( filename, mode ) ;
      if ( fp == NULL ) {
            fprintf(stderr, "Can not open %s\n", filename) ;
            exit( 2 ) ;
      }
      return( fp ) ;
}

void usage( char *name )
{
      fprintf(stderr, "usage: %s truthTable d1 d2 d3 d4 { outfile }\n",
            name) ;
}

int script( char *string )
{
      return( fprintf(ofp, string) ) ;
}

void GetOpt(int argc, char *argv[])
{
      int i ;
      ofp = stdout ;
      if (argc <= 5) {
            usage( argv[0] ) ;
            exit( 0 ) ;
      }
      if (argc > 5) {
            sscanf(argv[1], "%lu", &truthTable) ;
            sscanf(argv[2], "%d", &a[0]) ;
            sscanf(argv[3], "%d", &a[1]) ;
            sscanf(argv[4], "%d", &a[2]) ;
            sscanf(argv[5], "%d", &a[3]) ;
            for (i = 0 ; i < 4 ; i++ ) {
                  d[i] = a[i] ;
                  if ( d[i] < 0 ) {
                        d[i] += ((-d[i]/64) + 1) * 64 ;
                  }
                  d[ i ] %= 64 ;
```

FIG 6A

600

```
            }
        }
        if (argc > 6) {
            ofp = efopen(argv[6], "w") ;
        }
}

void printHead()
{
        script("#include <stdio.h>\n") ;
        script("#include <sys/types.h>\n") ;
        script("#include <sys/uio.h>\n") ;
        script("#include <unistd.h>\n") ;
        script("#include <fcntl.h>\n") ;
        script("#include <string.h>\n") ;
        script("\n") ;
        script("#ifndef BUFSIZ\n") ;
        script("#define BUFSIZ 4096\n") ;
        script("#endif /* BUFSIZ */\n") ;
        script("#define WSIZE 32\n") ;
        script("#define scmp(a,b) (strcmp((a),(b)) == 0)\n") ;
        script("#define rotateLeft(a,b) ((a<<b)|(a>>(WSIZE-b)))\n") ;
        script("\n") ;
}

void printMiddle()
{
        char string[ 256 ] ;
        script("\n") ;
        script("static unsigned long stateEven, stateOdd ;\n") ;
        script("unsigned long *returnState = &stateEven ;\n") ;
        script("\n") ;
        script("char *outFile = \"bsrand.32\" ;\n") ;
        script("unsigned long initUpper = 0x80000000 ;\n") ;
        script("unsigned long initLower = 0x00000000 ;\n") ;
        script("long initRun = 64 ;\n") ;
        script("long numData = 3000000 ;\n\n") ;
        script("void bsinit(unsigned long upper, ") ;
        script("unsigned long lower)\n{\n") ;
        script("\tint i ;\n") ;
          script("\tstateEven = 0 ;\n") ;
          script("\tstateOdd  = 0 ;\n") ;
        script("\tfor ( i = 0 ; i < 32 ; i += 2 ) {\n") ;
        script("\t\tstateEven <<= 1 ;\n") ;
        script("\t\tstateOdd  <<= 1 ;\n") ;
        script("\t\tstateEven |= (upper >> (31 - i)) & 0x01 ;\n") ;
        script("\t\tstateOdd  |= (upper >> (30 - i)) & 0x01 ;\n") ;
        script("\t}\n") ;
        script("\tfor ( i = 0 ; i < 32 ; i += 2 ) {\n") ;
        script("\t\tstateEven <<= 1 ;\n") ;
        script("\t\tstateOdd  <<= 1 ;\n") ;
        script("\t\tstateEven |= (lower >> (31 - i)) & 0x01 ;\n") ;
        script("\t\tstateOdd  |= (lower >> (30 - i)) & 0x01 ;\n") ;
        script("\t}\n") ;
        script("}\n\n") ;
        sprintf(string, "/* { %d, %d, %d, %d } */\n", a[0], a[1], a[2], a[3]) ;
        script( string ) ;
        script("unsigned long bsrand()\n") ;
        script("{\n") ;
        script("\tunsigned long newStateEven ;\n") ;
}
```

FIG 6B

600

```
void printTail()
{
      script("\tstateEven = newStateEven ;\n") ;
      script("\treturn( *returnState ) ; \n") ;
      script("}\n\n") ;

      script("void usage(char *name)\n{\n") ;
      script("\tfprintf(stderr, \"usage: %%s \", name) ;\n") ;
      script("\tfprintf(stderr, \"[-o outFileName] ") ;
      script("[-seed upper lower] \") ;\n") ;
      script("\tfprintf(stderr, \"[-even | -odd] [-help] \") ;\n") ;
      script("\tfprintf(stderr, \"[-num numData] " ) ;
      script("[-skip numInitialRun]\\n\") ;\n") ;
      script("}\n\n") ;

      script("void getOpt(int argc, char *argv[])\n{\n") ;
      script("\twhile(argc > 1) {\n") ;

      script("\t\tif ((argc > 2)&&scmp(argv[1], \"-o\")) {\n") ;
      script("\t\t\toutFile = argv[2] ;\n") ;
      script("\t\t\targc -= 2 ; argv += 2 ;\n") ;
      script("\t\t} else if ((argc > 2)&&scmp(argv[1], \"-num\")) {\n") ;
      script("\t\t\tsscanf(argv[2], \"%%ul\", &numData) ;\n") ;
      script("\t\t\targc -= 2 ; argv += 2 ;\n") ;
      script("\t\t} else if ((argc > 2)&&scmp(argv[1], \"-skip\")) {\n") ;
      script("\t\t\tsscanf(argv[2], \"%%ul\", &initRun) ;\n") ;
      script("\t\t\targc -= 2 ; argv += 2 ;\n") ;
      script("\t\t} else if ((argc > 3)&&scmp(argv[1], \"-seed\")) {\n") ;
      script("\t\t\tsscanf(argv[2], \"%%ul\", &initUpper) ;\n") ;
      script("\t\t\tsscanf(argv[3], \"%%ul\", &initLower) ;\n") ;
      script("\t\t\targc -= 3 ; argv += 3 ;\n") ;
      script("\t\t} else if (scmp(argv[1], \"-odd\")) {\n") ;
      script("\t\t\treturnState = &stateOdd ;\n") ;
      script("\t\t\targc-- ; argv++ ;\n") ;
      script("\t\t} else if (scmp(argv[1], \"-even\")) {\n") ;
      script("\t\t\treturnState = &stateEven ;\n") ;
      script("\t\t\targc-- ; argv++ ;\n") ;
      script("\t\t} else {\n") ;
      script("\t\t\tusage( argv[0] ) ;\n") ;
      script("\t\t\texit( 1 ) ;\n") ;
      script("\t\t}\n\t}\n}\n\n") ;

      script("int main(int argc, char *argv[])\n") ;
      script("{\n") ;
        script("\tint fd ;\n") ;
      script("\tunsigned long num, i, j, unit ;\n") ;
        script("\tunsigned long *buf ;\n") ;
        script("\tunit = BUFSIZ / sizeof( num ) ;\n") ;
        script("\tbuf = (unsigned long*)malloc( BUFSIZ ) ;\n") ;
      script("\tgetOpt( argc, argv ) ;\n") ;
        script("\tfd = open( outFile, O_WRONLY | O_CREAT, 0666 ) ;\n") ;
        script("\tbsinit( initUpper, initLower ) ;\n") ;
        script("\tfor ( i = 0 ; i < initRun ; i++ ) {\n") ;
        script("\t\tbsrand() ;\n") ;
        script("\t}\n") ;
        script("\tfor ( i = 0, j = 0  ; i < numData ; i++ ) {\n") ;
      script("\t\tbuf[j++] =  bsrand() ;\n") ;
        script("\t\tif ( j == unit ) {\n") ;
      script("\t\t\twrite(fd, buf, BUFSIZ) ;\n") ;
      script("\t\t\tj = 0 ;\n") ;
```

FIG 6C

```
      script("\t\t}\n") ;
        script("\t}\n") ;
        script("\tclose( fd ) ;\n") ;
      script("}\n") ;
}

void cleanUp()
{
      fclose( ofp ) ;
}

/*****************************************
 * displace.c
 *
 * Created: 2-Jul-2001
 * Author: Motoo Tanaka
 * Copyright (c) Hewlett Packard Company
 *****************************************/

void nextEven( int d )
{
      if ( d ) {
            if ( (d / 2)%32 ) {
                  fprintf(ofp, "rotateLeft(%s, %2d)",
                  (d % 2) ? "stateOdd " : "stateEven", d / 2 ) ;
            } else {
                  fprintf(ofp, "%s",
                  (d % 2) ? "stateOdd " : "stateEven" ) ;
            }
      } else {
            fprintf(ofp, "stateEven") ;
      }
}

void nextOdd( int d )
{
      if ( d ) {
            if ( ((d + 1) / 2) % 32 ) {
                  fprintf(ofp, "rotateLeft(%s, %2d)",
                  (d % 2) ? "stateEven" : "stateOdd ", (d + 1) / 2 ) ;
            } else {
                  fprintf(ofp, "%s",
                  (d % 2) ? "stateEven" : "stateOdd ", (d + 1) / 2 ) ;
            }
      } else {
            fprintf(ofp, "stateOdd") ;
      }
}

void nextState( int d[] )
{
      int i ;
      fprintf(ofp, "\tnewStateEven = f(\n") ;
      for ( i = 0 ; i < 4 ; i++ ) {
            fprintf(ofp, "\t\t") ;
            nextEven( d[i] ) ;
            if ( i != 3 ) {
                  fprintf(ofp, ",") ;
            }
            fprintf(ofp,"\n") ;
      }
```

600

FIG 6D

600

```
        fprintf(ofp, "\t) ;\n") ;

        fprintf(ofp, "\tstateOdd = f(\n") ;
        for ( i = 0 ; i < 4 ; i++ ) {
              fprintf(ofp, "\t\t") ;
              nextOdd( d[i] ) ;
              if ( i != 3 ) {
                    fprintf(ofp, ",") ;
              }
              fprintf(ofp,"\n") ;
        }
        fprintf(ofp, "\t) ;\n") ;
}

/*****************************************
 * TruthTable to Logic
 *
 * Created: 2-Jul-2001
 * Author: Motoo Tanaka
 * Copyright (c) Hewlett Packard Company
 *****************************************/

void i2b( unsigned int data )
{
        int mask ;
        for ( mask = 0x8000 ; mask > 0 ; mask >>= 1 ) {
              if (data & mask) {
                    fprintf(ofp, "1") ;
              } else {
                    fprintf(ofp, "0") ;
              }
        }
}


void emit( int data )
{
        int mask ;
        if ( isFirst ) {
              isFirst = 0 ;
        } else {
              fprintf(ofp, "|" ) ;
        }
        fprintf(ofp, "(") ;
        for ( mask = 0x8 ; mask > 0 ; mask >>= 1 ) {
              if ( mask < 0x8 ) {
                    fprintf(ofp, "& ") ;
              }
              if ((data & mask)==0) {
                    fprintf(ofp, "~") ;
              }
              fprintf(ofp, "a%X ", mask) ;
        }
        fprintf(ofp, ")") ;
}

void t2l( unsigned int truthTable )
{
        unsigned int mask ;
        int i ;
        isFirst = 1 ;
```

FIG 6E

```
        fprintf(ofp, "/* Logic for TruthTable %u */\n", truthTable) ;
        fprintf(ofp, "#define f(a8,a4,a2,a1) (") ;
        for ( i = 15 , mask = 0x8000 ; mask > 0 ; mask >>= 1, i-- ) {
                if ( truthTable & mask ) {
                        emit( i ) ;
                }
        }                                                              600
        fprintf(ofp, ")\n") ;
}

void doIt()
{
        printHead() ;
        t2l( truthTable ) ;
        printMiddle() ;
        nextState( d ) ;
        printTail() ;
}

main(int argc, char *argv[])
{
        GetOpt(argc, argv) ;
        doIt() ;
        cleanUp() ;
        return( 0 ) ;
}

Software Implementation of Cellular Automata
Based Random Number Generator
J. Barry SHACKLEFORD et al.
HP Docket No. 10019023-1
```

FIG 6F